Multibrot Sets Rotation in the Complex Plane

Nathan Lenzini

April 2025

1 Introduction

A number of years ago, as part of a project assigned to me in a high school geometry class, I took an interest in fractal sets such as the Mandelbrot and Julia fractals. Because I had very little understanding of complex algebra at that time, the approach I took was largely computational. This was fine, of course; a purely computational approach was more than enough to satisfy my curiosity at the time and generate some pretty satisfying images.

But I decided I wanted to revisit the topic now with the perspective of a little more math experience. In particular, I wanted to take a closer look at the concept of "Multibrot" sets, in which the exponent in the equation that governs the set has been increased. Doing this always created this cool, rotationally self-similar image, and I never understood why.

2 Math

I think many are probably pretty familiar with the concept of the Mandelbrot set. It's governed by the equation

$$f_c(z) = z^2 + c$$

Or, in a "Multibrot" set,

$$f_c(z; \alpha) = z^\alpha + c$$

where α is some real number and z is complex. A point c_0 in the complex plane is in the set if the recursive evaluation of $f_{c_0}(z)$ starting at $z = 0$ does not diverge to infinity. In practice, a finite threshold is used.

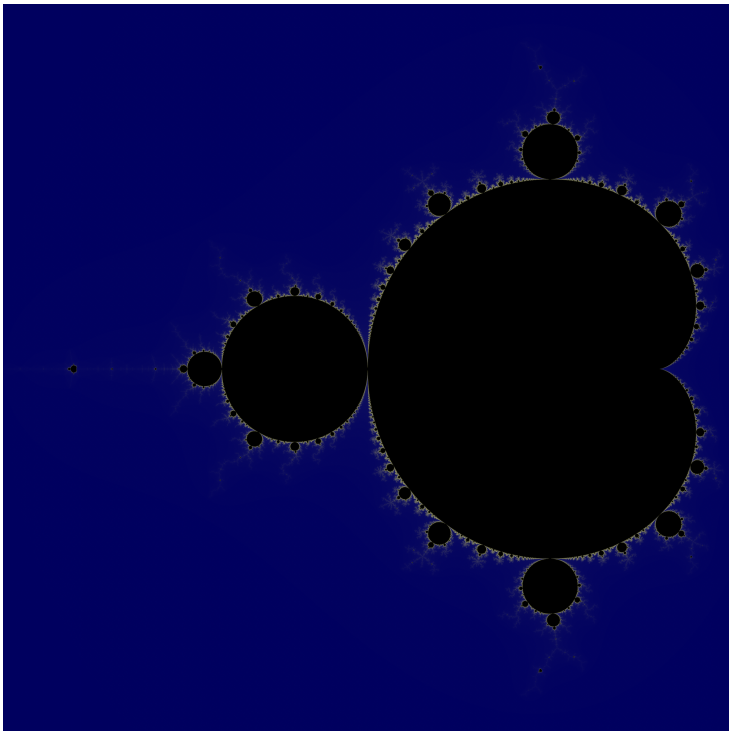

Figure 1: A rendering of the Mandelbrot set I made in 2020.

Of course, the fun rotational properties of the Multibrot sets come as a direct consequence of the analogous relationship between multiplication and in the complex plane.

3 Implementation

I decided to reimplement my fractal generation software (I've also learned some more about coding since 2020) and create an animation of the Multibrot set as α varies to illustrate the rotational behavior visually.

Originally, the code ran synchronously in Python. It opened and wrote to a file after checking each point in the domain, and then in a second pass did the same thing to render an image. Lots of optimizations were possible! I rewrote the code in Java and leveraged parallel processing techniques to speed up the set membership testing and image rendering. This allowed me to render several hundred frames in a very manageable time.

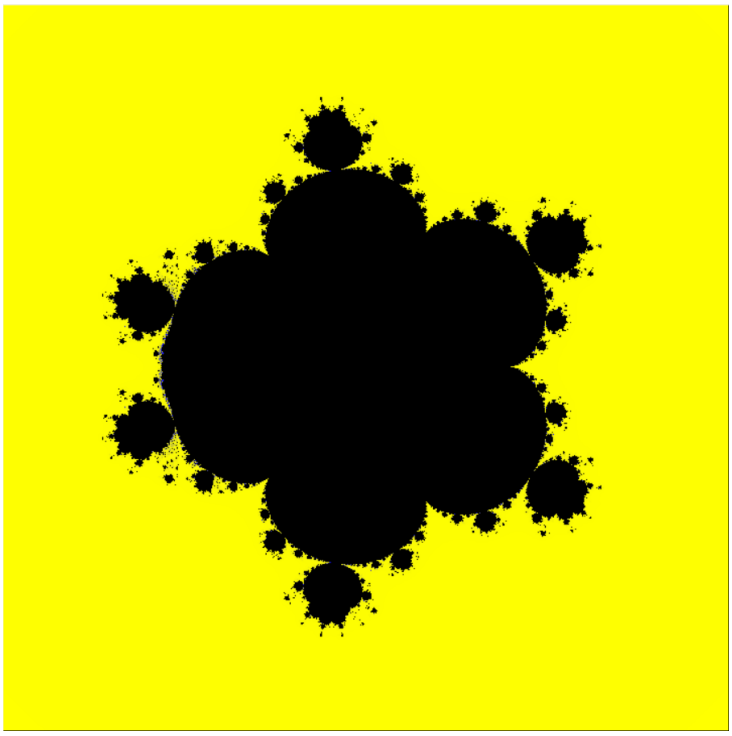

Figure 2: A rendering of a Multibrot set I made in 2020.

The resulting video is attached along with this submission. My code can be found on Github, at https://github.com/Amenhotep314/fractal_revenge.